On the Use of Reservoir Computing in Popularity Prediction

Tingyao Wu, Michael Timmers, Danny De Vleeschauwer, Werner Van Leekwijck
Alcatel-Lucent, Bell Labs
Copernicuslaan 50,
B-2018, Antwerp, Belgium
e-mail: {tingyao.wu, michael.timmers, danny.de_vleeschauwer, werner.van_leekwijck}@alcatel-lucent.com

***Abstract*— Predicting the life cycle and the short-term popularity of a Web object is important for network architecture optimization. In this paper, we attempt to predict the popularity of a Web object given its historical access records using a novel neural network technique, reservoir computing (RC). The traces of popular videos at YouTube for five continuous months are taken as a case study. We compare RC with existing analytical models. Experimental results show that RC, given a 10-day trace composed of daily cumulative views for a video, is able to predict the next-day's popularity with less than 5% relative square errors (RSEs). It is also demonstrated that RC achieves the best prediction performance among all compared models in longer-term prediction. The advantages and limitations of using RC in popularity prediction are discussed.**

Keywords - popularity prediction; reservoir computing; YouTube; popularity evolution

I. INTRODUCTION

Internet developed from a simple file exchanging network to a complicated, interactive network containing gigantic information levels. With the explosive evolution of the Internet, quickly identifying the popularity of online content is becoming valuable as popularity represents people's attention to online content. It is well known that the distribution of attention is far from balanced: most content receives little attention, while a small part of content gets a big proportion of attention [1][2]. Due to this asymmetry, service providers could benefit from early popularity prediction by using these predictions to better organize the content and apply optimized delivery strategies. For example, if content is predicted to be viewed many times in the near future, the provider may move it into a location with low latency and high-bandwidth access, improving user experience when retrieving the content.

It is observed that the popularity of online content can be determined soon after it is introduced. For instance, an advertisement or a video, which is accessed only dozens of times on its first days would be unlikely to become a big hit in the following days; however if more than a few hundred users access the content in the beginning, it might indicate to be a success item in the future. For these reasons, content popularity analysis has already become a hot research topic, branching into two subtly different directions. In one direction, researchers attempt to model the popularity distribution, also called popularity skewness, by looking at the statistics of a large database containing access traces and their ranks (e.g., [2][3][9]). In the other direction, researchers focus on an individual access trace, aiming at predicting how popular a Web object will be based on the access behavior during its early age. This is the task that we are addressing in this paper. The task is actually to predict the number of accesses, denoted as $\hat{y}_f(t)$, $t>n$, to online content at a future snapshot time t, given the historical access records of the content $y_h(t)$, $1\leq t\leq n$, where n represents the most recent time up to now, i.e., the time at which the prediction is made of how popular the object will be. There have been several analytical models proposed for this task, including linear correlation, log-transformed linear correlation, power-law or exponential decay model [2][4][5].

In this paper, we propose to use a novel neural network technique, reservoir computing (RC), to address this problem. Reservoir computing is derived from recurrent neural networks (RNNs) but avoids the well-known limitations of RNN, such as iterative parameter optimization and the condition of convergence. It has shown to excel in time series prediction [10], pattern classification [13], event detection [14], etc. Theoretically it can approximate non-linear dynamical systems with arbitrarily precision. One of its attractive advantages, its fading memory, distinguishes it from analytical models that usually assign the same importance to every historical observed access no matter how far they occurred in the past. Intuitively, the attention a Web object received recently should have a larger impact on the future than what happened in the beginning of its appearance. For instance, an online video that did not get many views after it was introduced on the server but receives a lot of attention recently probably will also continue the high attention for a while.

Popular videos at *YouTube* are taken as a case study to compare the different methods for popularity prediction. Established in 2005, *YouTube* is the world's largest user generated content (UGC) video system, and has already become one of the most successful Internet portals providing a short video sharing service. It contributes about 20% of all HTTP traffic or close to 10% traffic volume over the Internet, and this is expected to further increase [3]. On the other side, 10% of videos at the YouTube portal stand for 80% of total views [2]. Thus studying the popularity evolution for YouTube popular videos would give us an in-

978-0-7695-4185-3/10 $26.00 © 2010 IEEE
DOI 10.1109/INTERNET.2010.13

depth understanding on the life cycle of videos and specifically the impact to the Internet traffic.

The paper is organized as follows. In Section II, we present related studies in popularity prediction. In Section III we briefly introduce the theory of reservoir computing. The experimental design and results of using reservoir computing for popularity prediction are given in Section IV. The conclusions are presented in Section V.

II. RELATED WORK

With the development of Web 2.0, UGC based portals attract huge crowds. In this context, on the client side, popularity estimation helps to reduce the access latency; it is also important to the author(s) of the content. For instance, for advertisers who want to know the popularity of their video commercials or want to piggy-back their ads on popular content of others, as the online publication space is a highly competitive domain [4]. Two content sharing portals, *Digg* (www.digg.com) and *YouTube* (www.youtube.com) are usually studied as the examples of UGC based portals.

Usually there are two subtle research diversities. One research direction for popularity is to statistically analyze the relationship between the ranks of videos and their number of views, in order to discover the access patterns and the popularity of videos. Initially the Zipf law was used to describe the popularity distribution of videos, but quite a few alternative distributions have been proposed, including the power-law distribution with exponential cut-off tail [3], the Zipf-Mandelbrot distribution [6], the stretched exponential distribution [7], the log-logistic distribution (for normal videos) [9], the Weibull distribution (for popular videos) [9], etc.

The other research direction for online content popularity is to predict the number of accesses an object will receive in the future given the access patterns at a time just after its introduction time. In [8], with the incorporation of comment data and a co-participation network, the popularity of links in *Digg* was studied. The authors were able to predict the popularity of news linked at *Digg* shortly after the introduction time of the news item. In [2], although not really predicting the popularity, the authors found that a linear correlation existed between the number of views on the 2nd day and the 90th day of the birth (the correlation coefficient was 0.84) after analyzing about 11,000 YouTube videos in the "*sci*" and "*ent*" categories. Szabo and Huberman [4] presented a similar analysis of the near-future popularity prediction by investigating the life cycle of links on *Digg* and videos on *YouTube*. In their YouTube video study, they collected the daily views of more than 7,000 "recently added" videos for one month. They observed a stronger linear correlation between the log-transformed popularities at an early time and a later time: the correlation coefficients between the cumulative views on day 7th and day 30th were 0.77 and 0.92 before and after the logarithmic transformation respectively. Motivated by the fact that most of popular videos follow the power-law distribution, but also have a heavy tail, Avramova et al. [5] proposed a non-linear analytical model to cover both cases, which could be degenerated into a power-law or exponential decay law depending on the choice of parameters. The authors demonstrated the validity of the model in several experiments. In [15], Crane and Sornette studied the relaxation response after endogenous and exogenous bursts of activities for the daily views of YouTube videos, and classified the popularity evolution of online videos into three categories. It was clearly shown that one category exposed a significant precursory growth followed by an almost symmetric relaxation and the other two categories showed a sudden burst of activity followed by a power-law relaxation.

III. RESERVOIR COMPUTING

In this paper, we use reservoir computing to predict the popularity of popular YouTube videos. On top of its proven excellent performance on time series prediction, RC employs fading memory which fits the characteristics of popularity.

Reservoir computing, independently proposed by Jaeger [12] and Maass [11], is derived from recurrent neural networks but it is also a drastically different approach to RNN. First, it greatly simplifies the RNN training approach, avoiding using the gradient descent method to look for a local optimal solution. Second, it does not require the procedure of global parameters optimization; so the computational cost is quite trivial. Finally, it has already been shown (e.g., [10]) that RC works well on a lot of learning relevant tasks.

A typical RC scheme is depicted in Fig. 1. As illustrated, the weights between the input and the neurons in the reservoir $\mathbf{W}_{inp}^{res}$ and the internal weights between the neurons $\mathbf{W}_{res}^{res}$ are randomly chosen and then fixed. To guarantee the stability of the reservoir, $\mathbf{W}_{res}^{res}$ must be scaled so that the largest eigen- value $|\lambda|_{\max}$ and the spectral radius r of $\mathbf{W}_{res}^{res}$ satisfy $|\lambda|_{\max} \le 1$ and $0 \le r < 1$. The temporal state in the reservoir is computed as:

$$x(t) = f(\mathbf{W}_{inp}^{res} u(t) + \mathbf{W}_{res}^{res} x(t-1)) , \qquad (1)$$

where $u(t)$ is the input signal and the function f is a non-linear activation function of the reservoir neurons, typically the tanh(.) function. In order to reduce the effects of the arbitrary starting state and make sure that the network states are a pure reflection of the input signal, some warm-up states in the beginning part of $x(t)$ need to be discarded before the training. The factor, determining the percentage of the warm-up states, is called the warm-up factor. In the readout part, given the temporal teacher signal $y_h(t)$ (also shown in Fig.1) , only the weights between the reservoir and the output $\mathbf{W}_{res}^{out}$, and optionally, the weights between the input and the output

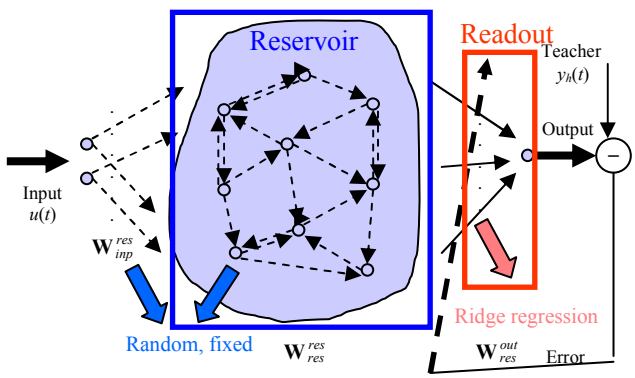

Figure 1. The topology of reservoir computing

$\mathbf{W}_{inp}^{out}$, and the bias weights $\mathbf{W}_{bias}^{out}$ (not shown in Fig. 1) are trained based on the truncated reservoir states $x(t)$, typically using ridge regression. Then the predicted output signal $\hat{y}_f(t)$ at time t ($t>n$) is computed as:

$$\hat{y}_f(t) = f(\mathbf{W}_{res}^{out} x(t) + \mathbf{W}_{inp}^{out} u(t) + \mathbf{W}_{bias}^{out}) \qquad (2)$$

IV. Experiments

In this section, we will first describe the collection of our database. The evaluation criterion is then introduced, followed by a discussion of different models used to benchmark our technique. Afterwards, we present the prediction performance of the RC approach and compare it with the other models. An analysis of the experimental results is discussed at the end of this section.

A. Data collection

We use the same YouTube database as described in [5]. The 50 most popular videos announced on the USA's (denoted as US dataset) and Japanese (denoted as JP dataset) YouTube websites were daily monitored from 20th July to 31st December, 2008 using the APIs provided by *YouTube*. The accumulative number of views for every monitored video was tracked once each day for each monitored video. As every day different popular videos were announced, the number of monitored videos was increasing and the longest trace is 160 days. However, on certain days the cumulative views could not be retrieved for some videos, due to unavailability of the video itself or its statistics, leading to incomplete traces with significant gaps. As a result, although in total approximately 10,000 videos were monitored, only 1,728 and 1,445 of those traces have more than 30 retrieved entries in the US and JP datasets respectively. Note that the number of applicable traces in our experiments is larger than those used in [5], due to the inclusion of the traces with significant gaps. Furthermore, note that only when a video was listed in "popular videos" category, its URL was identified and subsequently monitored. The evolution before it arrived in this list is, hence, not known. For each monitored video, there is list of pairs ($u(t)$,$y(t)$) which have the following interpretation: $u(t)$ days after the start of the monitoring of that particular video, it had had $y(t)$ cumulative accesses. Hence, $y(t)$ represents the cumulative popularity. In order to obtain the momentary popularity $y(t)$ has to be derived with respect to $u(t)$: i.e., the momentary popularity in terms of access rate between two consecutive recording days is $(y(t)-y(t-1))/(u(t)-u(t-1))$. A trace $y(t)$ is divided at a certain snapshot n into two parts: the first part is $y_h(t)$ ($t \leq n$), used as the training trace, and the second part $y_f(t)$ ($t>n$), used as the test trace.

B. The criterion for performance evaluation

We evaluate the performance of all models with different lengths of training traces. The number of training entries is set to be 10, 20, 30, 40, 50, 90 and 120 days. As we are more interested in the near-future prediction, unless stated otherwise, the length of a test trace is arbitrarily set to be 10 days, regardless of the length of the training trace, meaning that we will predict the popularity of a video for the next 10 days. We take the relative squared error (RSE) criterion to evaluate the prediction performance [4]. Given training trace s with κ training entries, the performance on a future day $\kappa+\tau$ is evaluated as the relative squared error between the prediction and the actual popularity:

$$RSE^s(\kappa,\tau) = \left(\frac{\hat{y}_f^s(\kappa+\tau) - y_f^s(\kappa+\tau)}{y_f^s(\kappa+\tau)} \right)^2 \times 100\% , \qquad (3)$$

We also use this notation for the ensemble average values too: $RSE(\kappa,\tau) = < RSE^s(\kappa,\tau) >_s$, where <•> indicates the average across all training traces.

C. Models of popularity prediction

In this section, we present the realization of all modeling approaches used in our experiments. For some models, a parameter optimization procedure is needed. As a result we use the US dataset as the training set to obtain the required parameters and use the traces in the JP dataset for prediction.

1) Linear correlation

In [2], a high linear correlation (denoted as *LIN*) between the cumulative views at two snapshots has been claimed. Following this model, the correlation coefficients ρ_{ij} for any pair of days (i, j) are computed over the US dataset. In prediction, given a training trace s with κ entries in the JP dataset, to predict the popularity on day $\kappa+\tau$, we select day d in the training trace whose correlation coefficient with the predicted day is maximum: $d = argmax_{\forall i \in \{1,\ldots,\kappa\}}(\rho_{i,\kappa+\tau})$ (in fact in more than 95% cases d is the latest training day). Then a linear regression model is built based on the popularities on day d and the popularities on day $\kappa+\tau$ for all applicable traces in the US dataset. The cumulative views on day $\kappa+\tau$ is predicted based on the regression formula and the cumulative views on day d.

2) Log-transformed linear correlation

A linear correspondence between the log-transformed popularities has been discovered in [4]. The way to compute the correspondence γ_{ij} between the popularities on the pair of days (i, j) is to minimize RSE on the US dataset: $\gamma_{ij} = \frac{\sum_s y^s(i)/y^s(j)}{\sum_s (y^s(i)/y^s(j))^2}$. The popularity on day $\kappa+\tau$ is then predicted as

$$\hat{y}_f^s(\kappa+\tau) = < e^{\gamma_{t,\kappa+\tau} \log(y_h^s(t))} >_t . \quad (4)$$

We denote this method as *LogLIN_AVE*. An improved prediction method following the same idea, called *LogLIN_C10*, takes the most recent 10 training entries into consideration ($t = \kappa$-9, κ-8,…, κ in (4)). This model assumes that the near-future popularity only depends on the recent access history.

3) Power-law versus exponential dcay function (PLED)

In [5], the authors modeled the cumulative distribution of a trace with:

$$\hat{y}_h(t) = \rho \left[1 - \left(1 + \frac{\left(\beta^{-\frac{1}{\alpha}} - 1 \right)(t-\theta)}{\tau} \right)^{-\alpha} \right] \quad (5)$$

This model is denoted as *PLED*. We use the Nelder-Mead simplex method to optimize the parameters. Note that with this method, the US dataset is not required.

4) Reservoir computing

The RC approach is implemented using the reservoir computing toolbox v2.0 [16]. We use the sequence of day index as the input for the network and the sequence of cumulative views as the teacher signal. In the test session, the input is the day index on which we want to perform prediction, and the output is the predicted popularity. A fully-connected RC is used where the internal weights are uniformly randomized between [-1, 1]. Two unknown parameters, the size of the reservoir and the warm-up factor, are optimized using the US dataset as we wonder which topology of RC is the most suitable one for this task. The size decides the capacity of the reservoir whereas the warm-up factor balances the randomization effect and the remaining number of training states. We will also investigate the effect of the randomness to our prediction precision in the experiments. However, as shown in the experiment, even if we discard the first 30% of training entries attempting to erase the influence of randomness, the prediction accuracy is still fluctuating, due to the limited number of training entries for one trace. Thus in each single prediction, the RC is repeated 30 times, and we take the average of the 10 instances of which the means are in the middle as the predicted value on day κ+τ.

TABLE I. CORRELATION COEFFICIENTS OF VIDEO VIEWS IN TWO TIME SNAPSHOTS ON THE US DATASET

Original domain/ Log-transformed domain			
***Early day* (x_0)** [2]	***X_0 +5 days***	***X_0 +7 days***	***X_0 +90 days***
Day 2	0.86 / 0.94	0.83 / 0.92	0.37 / 0.78
Day 3	0.94 / 0.97	0.92 / 0.96	
***Early day* (x_0)** [4]	***X_0 +23 days***		
Day 7	0.84 / 0.95		

D. Experiments

1) Correlation inspection

Although only when a video became popular it was tracked, we still see the existence of strong correlations between the popularities or the log-transformed popularities on an early day and a later day in our database, as claimed in [2] and [4]. The correlation coefficients in two example time snapshots on the US dataset are presented in Table I. We do observe that the correlation in the log-transformed domain is stronger than the one in the original domain, similar to the observation in [4].

2) Parameter optimization for RC

We use the US dataset to look for the appropriate size of the reservoir and the warm-up factor. Five sizes of the reservoir (1, 4, 10, 40 and 100) and three potential warm-up factors (10%, 20% and 30% of training entries) are attempted. The searching is done on a PC with Intel Core 2 Duo CPU P8400@2.26GHz and the simulation spent about 1.2 hours in Matlab 7.0 environment. The RSEs of the RC over different lengths of training entries with different parameters on the US database are shown in Fig. 2. As can be seen, in most cases, the performance with the 30% warm-up factor is slightly better than with the other two factors. On the other hand, building the reservoir with more than 10 neurons is not worthwhile, as the prediction error rate does not decrease significantly with the increasing size. So in the following experiments, the size of the reservoir and the warm-up factor are set to be 10 and 30% respectively.

3) Performance comparison

Fig. 3 presents the RSEs for all compared models with different number of training entries on the JP dataset. The horizontal axis represents the day gap τ between the predicted day and the last training day. We can see that, first, the *LIN* approach is not appropriate for the popularity predicting task as its prediction error rate increases rapidly along with the day gap. Second, the *LogLIN_AVE* approach is much less effective than *LogLIN_C10*, although both follow the same transformation. For the former one, the RSE is about 20% regardless of the length of training traces, while the latter method could obtain less than 5% averaged RSE when κ=30. Third, if we compare the three best prediction approaches, RC, *PLED* and *LogLIN_C10*, we can see that RC uniformly outperforms the other two when the training traces are longer than 30; whereas in short training cases,

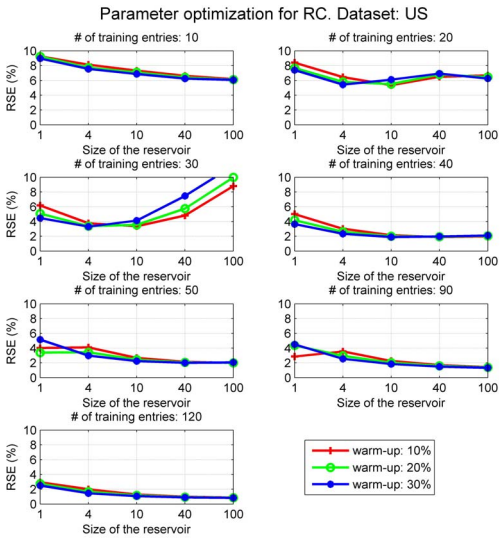

Figure 2. Parameter optimization for the RC approach.

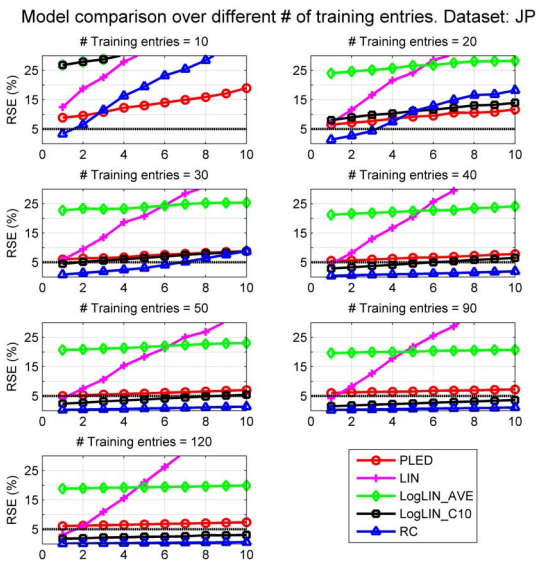

Figure 3. The average relative errors (%) of all models for predicting the next 10-day cumulative views on the JP dataset

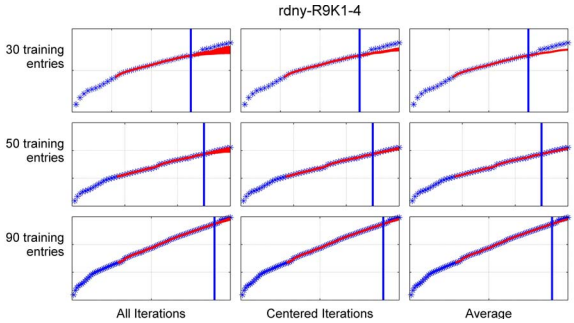

Figure 4. An example of the randomization effects with RC.

such as 10 and 20, RC takes the lead in the very close future but is less accurate in the moderate future. Finally, not all methods benefit from the increasing number of training entries. For example, with the RC approach, the RSEs for predicting the next day's popularity with 10 and 120 training entries are 3.37% and 0.12% respectively. But for *PLED* and *LogLIN_AVE*, the prediction accuracies are almost saturated when κ is over 40.

E. Randomization effects in the RC approach

Although the warm-up factor discards the first 30% of reservoir states to diminish the influence of randomness, we still can see the randomization effects in the prediction. Fig. 4 shows an example about how the trace of the cumulative views of a video is predicted by the RC. The horizontal and vertical axes are the day counting from the start of recording, and the corresponding popularity respectively. From the left to the right, we plot the fitted curves for all 30 iterations, for the middle 10 iterations and the average of the middle 10 iterations in turn. From the top to the bottom, a few different lengths of training entries are tried. The vertical blue line indicates the boundary of the training entries and the predicted entries. In the case of short training entries, the weight randomness leads to a large diversity of predicted entries, which can be reduced by selecting the predictions whose means are in the middle. On the other hand, the more training entries, the more stable the prediction will be.

F. Number of known entries vs. prediction accuracy

Fig. 3 illustrates the RSEs of different models over various lengths of training traces. Especially, for the RC approach, the prediction error monotonically decreases with the increasing number of training entries. It would be interesting to investigate that given a set of training entries, how many days in the future that we can assure to accurately predict the popularity. Hence, we define 5% as the threshold: if the RSE of a predicted day is less than this threshold, we assume the day is correctly predicted. Note that in this case the length of test entries is not constraint to 10 anymore. As shown in Fig. 5, the RC could provide 1-, 5- and 20-day good predictions for 10, 30 and 40 training entries respectively, which are superior to other models.

G. Discussions

In Table II, we conclude the pros and cons for the RC approach together with other good popularity modeling methods. The RC shows excellent prediction performance in this task with a simple topology and fading memory. However, as it is a black box, its prediction capability is difficult to analytically interpret; moreover, because it is usually required to predict the popularity in an early age, the randomization effect still exists in the prediction. However the effect can be neutralized by smoothing.

In general, the *PLED* method models the whole evolution

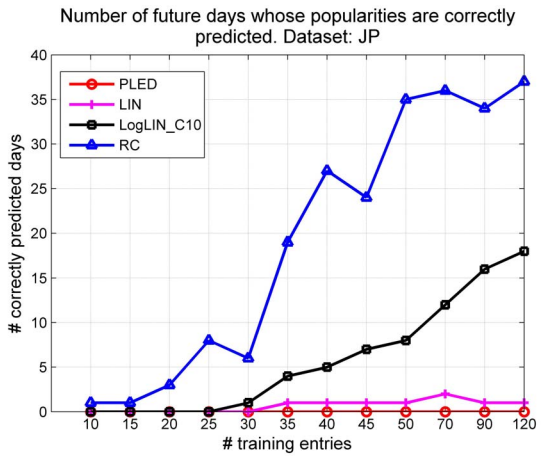

Figure 5. Number of future days whose popularities are correctly predicted (RSE<5%) with different lengths of training entries.

TABLE II. PROS AND CONS FOR RC AND OTHER MODELS

	RC	***PLED***	***LogLIN_C10***
Few training entries	Excellent	Excellent	Bad
Many training entries	Excellent	Saturated at 30 entries	Excellent
Importance for each entry	Different	Same	Different
Interpretability	Black box	Yes	Yes
# of parameters	~10	4	~100

of video popularity quite well and is analytically interpretable. Nevertheless, because it assigns the same importance to each observed training entry, the model has to look for a balance between the recent popularities and the historical ones. Thus it does not perform best in our prediction experiments.

We also see that the prediction based on the linear correlation between the popularities in an early age and a later age is not reliable, as shown in Fig. 3. On the other hand, the log-transformed popularities do have stronger correlation, thus the two relevant approaches, especially *LogLIN_C10*, show the good prediction precision. For example, predicting the popularity on the 31st day based on the records between 21st – 30th days results in a 4.31% prediction error, which is similar as the results presented in [4].

Finally, because of the limitation of the data collection, we could not investigate the popularity evolution for unpopular videos and videos with sudden bursts. These are the scenarios that we will explore in the future work.

V. CONCLUSIONS

In this paper, we presented the study of using reservoir computing in popularity prediction. We compared its prediction performance with other prediction models based on the study of YouTube popular videos. It was shown that the prediction is not reliable where it is merely based on the linear correlation between two (log-transformed) popularities. The natural emphasis on recent popularities makes reservoir computing successfully predict the popularity of a YouTube video the next day given 10-day records, or the next 5 days given 30 historical accesses. We also investigated the effects of randomness in this task and employed a smoothing way to wash out the randomness.

ACKNOWLEDGMENT

This work was partially supported by the European project: Open ContEnt Aware Networks (OCEAN), grant agreement no. 248775.

REFERENCES

[1] F. Wu and B. A. Huberman. "Novelty and collective attention," Proc. the National Academy of Sciences, 104(45), 2007, pp.17599-17601 .

[2] M. Cha, H. Kwak, P. Rodriguez, U-Y, Alm and S. Moon, "I Tube, you Tube, everybody Tubes: analyzing the world's largest user generated content video system," Proc. the 7th ACM SIGCOMM Conference on Internet Measurement, 2007, pp. 1-14.

[3] X. Cheng, C. Dale, and J. Liu. "Understanding the characteristics of Internet short video sharing: YouTube as a case study," Proc. the 7th ACM SIGCOMM Conference on Internet Measurement, 2007, pp. 28-36.

[4] G. Szabo, and B. Huberman, "Predicting the popularity of online content", available online at www.hpl.hp.com/research/scl/papers/predictions/predictions.pdf.

[5] Z. Avramova, S. Wittevrongel, H. Bruneel and D. De Vleeschauwer, "Analysis and modeling of video popularity evolution in various online video content systems: power-law versus exponential decay", Proc. Internet2009, pp. 95-100.

[6] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat, "Modeling and Generating Realistic Streaming Media Server Workloads", Computer Networks, vol. 51, 2007, pp. 336-356.

[7] L. Guo, E, Tan, S. Chen, Z. Xiao, and X. Zhang, "The stretched exponential distribution of Internet media access patterns", Proc. 27th ACM Symposium on Principles of Distributed Computing, 2008, pp. 283-294.

[8] S. Jamali and H. Rangwala, "Digging *Digg*: Comment mining, popularity prediction and social network analysis", Technical Report GMU-CS-TR-2009-7.

[9] A. Abhari and M. Soraya, "Workload generation for YouTube", Multimedida Tools Applications, vol. 46(1), 2010, pp. 91-118.

[10] H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication", Science, vol. 304(5667), 2004, pp. 78-80.

[11] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations", Neural Computing. vol. 14, 2002, pp. 2531–2560.

[12] H. Jaeger, "The echo state approach to analysing and training recurrent neural networks", (GMD-Report 148, German National Research Institute for Computer Science 2001).

[13] S. Klampfl and W. Maass. "Replacing supervised classification learning by Slow Feature Analysis in spiking neural networks". Proc. Neural Information Processing Systems, vol. 22, 2009, pp. 988-996.

[14] E.A. Antonelo, B. Schrauwen, and D. Stroobandt, "Event detection and localization for small mobile robots using reservoir computing", Neural Networks, vol. 21, 2008, pp. 862–871.

[15] R. Crane and D. Sornette, "Robust dynamic classes revealed by measuring the response function of a social system", Proc. the National Academy of Sciences of the United States of America, no. 41, 2008, pp. 15649-15653.

[16] http://reslab.elis.ugent.be/rctoolbox, latest access on 2nd June 2010.